ARL-SR-0411 ● Dec 2018

US Army Research Laboratory

A Primer on Machine Learning for Materials and Its Relevance to Army Challenges

by Brian C Barnes

ARL-SR-0411 ● Dec 2018

US Army Research Laboratory

A Primer on Machine Learning for Materials and Its Relevance to Army Challenges

by Brian C Barnes
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* December 2018	**2. REPORT TYPE** Special Report	**3. DATES COVERED (From - To)** 1 September 2018–1 December 2018
4. TITLE AND SUBTITLE A Primer on Machine Learning for Materials and Its Relevance to Army Challenges		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Brian C Barnes		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-WML B Aberdeen Proving Ground, MD 21005		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-SR-0411
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Machine learning and related topics (data science, artificial intelligence, etc.) are discussed in the context of their application to US Department of Defense weapons and materials problems. A brief introduction to this research area and discussion of a few common questions are provided for a general audience of scientists or engineers. Concise self-instruction guides are provided to educate researchers beginning work in the field who need to quickly get up to speed and execute professional research projects.

15. SUBJECT TERMS
machine learning, artificial intelligence, data science, materials, discovery

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Brian C Barnes
Unclassified	Unclassified	Unclassified	UU	20	**19b. TELEPHONE NUMBER (Include area code)** 410-306-0772

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

The US Army currently faces increasingly urgent demands to improve the performance of materials in its fielded weapons and armors. Advances in the field of machine learning (ML) and in high-performance computing allow for entirely new approaches to material discovery, data analytics, process optimization, and system design.[1–3] ML leverages the explosion of data available from high-resolution experiments and modern, high-throughput simulations and is being applied to sparse, scattered, historical datasets. Through integration of ML techniques in the R&D workflow, the Army could transition solutions to the Soldier more rapidly and at lower expense, and those solutions may be more effective.

The growing impact of ML on the US Department of Defense (DOD) was also recently recognized by Congress in the John S McCain National Defense Authorization Act for Fiscal Year 2019 (Section 238 on Joint Artificial Intelligence Research, Development and Transition Activities)[4] and by Gen John M Murray who said "that's exactly the type of people I need" at the 2018 Association of the US Army Annual Meeting[5] in reference to a Soldier with a PhD in data science.

One potential impact area is molecular materials. For example, the high-explosive hexanitrohexaazaisowurtzitane (CL-20) has great performance, yet it is still not widely fielded despite its discovery 30 years ago, in part because it cannot be produced in a cost-effective manner. Efforts at finding an equally effective, practical alternative to CL-20 have not yet been successful. The pharmaceutical industry, faced with similar challenges in discovery and production, embraced ML to accelerate their product pipelines.[6] Solar cell materials have been similarly discovered,[7] and this ML workflow may be applied to DOD materials.

ML for materials research is not limited to organics or pure materials. It has also been applied to inorganic materials,[8] including the study of microstructure[9] and brittle fracture,[10] which is a key issue faced by the Army for armor discovery and design. Development and fielding of next-generation propellants may strongly benefit from use of data-driven modeling in design of experiments for formulations where physics-based models are unavailable. ML can leverage "Big Data" sourced from experiments or high-fidelity simulations to aid analysis and discover entirely new materials or system designs through high-throughput virtual screening or architectures such as generative adversarial networks.[11,12] ML may reduce computational time to solution through on-the-fly learning of fast-running models.[13] This report will answer some common questions addressed to practitioners in the field, and end with self-instruction guides to build skills and perform work using these techniques.

1.1 What Do These Terms Mean?

Over the past several years, there has been rapid growth of work in the chemistry and materials literature applying "machine learning" techniques for the prediction of material properties, development of new simulation techniques, or complementing experiment in discovery of new materials or improved processes. This general type of work is also sometimes referred to as using "data science", "artificial intelligence" (AI), or "data mining" approaches. The following is an academic definition of ML, from "The Discipline of Machine Learning"[14] by Professor Tom Mitchell at Carnegie Mellon University's School of Computer Science:

> To be more precise, we say that a machine learns, with respect to a particular task T, performance metric P, and type of experience E, if the system reliably improves its performance P at task T, following experience E. Depending on how we specify T, P, and E, the learning task might also be called by names such as data mining, autonomous discovery, database updating, programming by example, etc.

Common usage of the term "machine learning" does not always follow that academic definition. Descriptively we may claim, based on how people use the terms in papers and on the Web (actually influential in this area), that data science produces insights, ML produces predictions, and AI produces actions (see http://varianceexplained.org/r/ds-ml-ai/ for an expanded discussion). Predictions may be of continuous-valued properties, such as the melting point of a material, in which case it is referred to as a regression model. Predictions may also select the most likely class among a number of discrete options, such as whether a pharmaceutical molecule is inactive, moderately active, or active in regard to inhibiting a particular virus. Problems regarding choice of class are referred to as classification problems. When some ground truth is known and used for training of a predictive model, it is referred to as "supervised" learning.

Actions produced by an AI model are generally a sequence of choices in response to an environment that changes over time due to external factors. The training process is often referred to as "reinforcement" learning. For example, the work on AlphaGo and later, AlphaGo Zero,[15] trained a neural network model able to play the game of Go without attempting to exhaustively enumerate possible future positions or rely on a look-up table of past actions by human grandmasters. Enumeration and look-up tables were common (and successful) strategies for early computer chess programs but less successful for the more complex game of Go.

In the chemistry literature, the choice to describe work as AI or ML is often due to the objective of the work or the use of certain classes of algorithms. For example, work using neural network algorithms may be referred to as AI despite their application to a regression or classification problem that could also be addressed through use of a support vector machine (another algorithm capable of discovering nonlinear relationships in data).[16,17] Unfortunately, this shows that usage of the terms does depend on the field in which they are being used. Leading works in the chemistry literature demonstrate a broad interpretation of machine learning, often tied to creating a predictive model.[18–26]

ML or AI model construction in this context is “data-driven”, meaning that the models typically do not assume any particular relationship between input variables and the output.[21] This is opposed to physics-driven or empirically motivated parameterized models. For example, the empirical Kamlet–Jacobs relations for prediction of detonation properties include that the scaling of an explosive’s detonation pressure increases with the square of its density. That may not be explicitly included in a neural network model; instead, the model could discover the dependence of detonation pressure on density during its training. A potential advantage of the data-driven approach is the ability to have improved predictive accuracy, perhaps at the cost of some interpretability (the ability to explain a model or result in terms understandable to a human). As stated by Shmueli,[27] “Explanatory power and predictive accuracy are different qualities; a model will possess some level of each”. Another perspective on model construction philosophy is provided by Breiman in his landmark article “Statistical Modeling: The Two Cultures”.[28] One of his comments is that by focusing on easily interpreted stochastic models with assumed forms (instead of embracing techniques such as random forests or neural nets), “statisticians have ruled themselves out of some of the most interesting and challenging statistical problems that have arisen out of the rapidly increasing ability of computers to store and manipulate data”. Both Shmueli and Breiman are from the statistics community; techniques underpinning much of today’s AI/ML work were developed by the statistics or computer science communities decades ago. The recent success and popularity of these techniques is due largely to four factors: 1) an increase in cheap and abundant computing power, 2) an increase in large, carefully curated datasets, 3) breakthroughs in algorithm development, and 4) maturation of fully automated workflows for processing, model development, and analysis of data.

Finally, we note that data-driven models are only as good as their input data; “garbage in, garbage out”. A large percentage of the time for a new ML research project may be spent in curation of a robust dataset. Some of the most highly cited works in ML research, such as the MNIST database of handwritten digits, are just painstakingly curated datasets.

1.2 How Does a Basic Neural Network Operate?

In Fig. 1, we show how a small feed-forward neural network operates. We assume that the model has already been trained: its weights **W** and biases **b** have been optimized to generate a useful model for the problem at hand. Input is a vector **x** of rank *i* (where *i* is the number of input nodes). Each vector element is simply a number. As input is passed to the hidden layer, it is multiplied by linear weights **W** for each edge. At each node in the hidden layer, incoming edge values are summed, a bias value is added, and then nonlinear function *h*(x) is applied to the resulting value. The choices of nonlinear function (such as a hyperbolic tangent, or "tanh") and number of hidden nodes (in Fig. 1, three green circles) is made during the model construction process and may be considered a hyperparameter to be optimized. The nonlinear functionality in the hidden nodes is what allows neural networks to be more useful than a simple linear regression model.

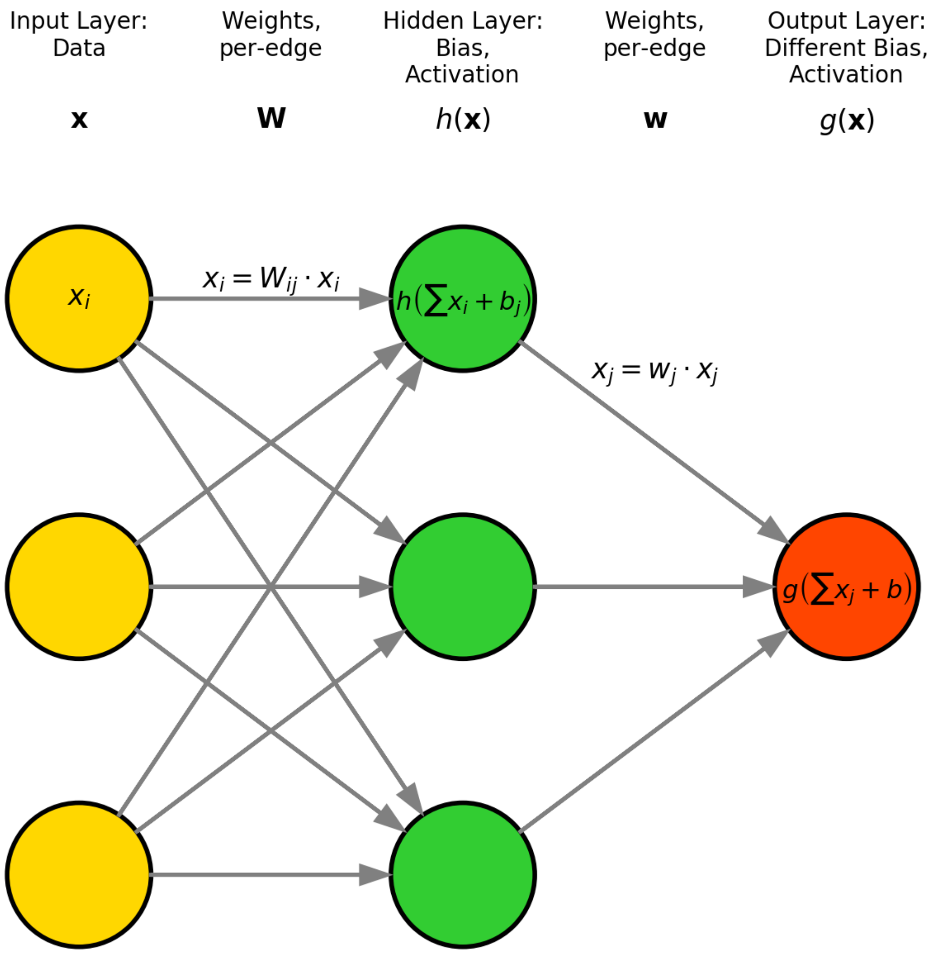

Fig. 1 Information flow of a small feed-forward neural network (also known as multilayer perceptron), drawn with nodes (circles) and edges (arrows). Input nodes are drawn in yellow, hidden nodes in green, and the output node in orange. Math operations are shown for a subset of the network.

Considering all hidden nodes, the intermediate values may be described as a new vector **x** of rank *j* (where *j* is the number of nodes in the hidden layer). These values are passed along a set of edges to the output layer and weighted in a similar fashion as before. At the output node, incoming edges are summed and a bias is added, and then a function *g*(x) is applied to the resulting value. This function is often different from the activation function used for hidden nodes, and may be linear. For classification problems, where the network is trying to select among limited options, *g*(x) is often a sigmoid or softmax function, and there may be many nodes in the output layer (one per class determination). For typical regression problems, where the network is trying to predict a continuous number, *g*(x) is a linear activation, and there is one node in the output layer. For large neural networks, the number of weights to be trained increases very quickly, and training may be a challenging, time-consuming process. A subject matter expert applying neural networks to their field may advance the state of the art by determining the best way to construct an input vector **x** (sometimes called a "featurization" or "descriptor") and the best type of neural network (or "topology") for their problem. For the purposes of this report, we limit our explanation of ML methods to this simple neural network. Neural networks have a wide variety of applications that are sometimes viewed as magical black boxes; we wanted to show some of the not-so-magical underpinning math. Neural networks are not the only algorithms to be used for ML of materials. Other common methods include, but are not limited to random forests,[29] Gaussian process regression,[30] and support vector machines.[31]

1.3 Is Machine Learning Just Interpolation and Only Useful When Interpolating Between Training Set Data Points?

The question has different answers depending on the problem being studied and the algorithm being used. Multilayer perceptrons may effectively be interpolating, but their optimal input representation may be very different from that used in linear regression. Neural networks introduce parsimonious, nonlinear transforms of input element combinations, so that the final hidden layer that is being interpolated over will be quite different from the input space. Given sufficient data, neural networks should have a different (hopefully lower) error when faced with an out-of-set example as compared with a linear regression model for a problem with a nontrivial underlying mechanism.[21,32] Kernel-based methods allow for analysis in a higher-dimensional space than the native representation of the data, which makes problems linearly separable that were previously inseparable.[33] In the case of regression problems, kernel methods conveniently allow for modeling of periodic time-varying effects and for nonlinear weighting of data points based upon a similarity metric (or data "distance"). This allows them to very effectively interpolate.

Random forests may discover key rules in classification problems that allow reliable extrapolation to unseen data if the pertinent underlying rule is represented in the training set.[29] Convolutional neural networks operating on 2-D/3-D pixel or voxel data may be interpolating by first "learning" localized, intermediate representations and multidimensional interactions that would be practically impossible to model by interpolating among training images.[34] Unsupervised learning does not leverage any "ground truth" dataset; therefore, is not interpolating among a reference set. Instead, it is usually used to discover relationships present but not described within a dataset. The most-common examples of unsupervised learning are clustering and dimensionality reduction techniques, and unsupervised methods are often used as part of a larger workflow for predictive modeling. Reinforcement learning (e.g., AlphaGo) determines *policy* (or control strategy) for an *agent* (or independent actor in an unknown dynamical environment), and this process requires more than value estimation for a state (so it is not just interpolation). Reinforcement models are often associated with research toward autonomously acting agents such as artificially intelligent robots or game-playing AI. These models are often referred to as AI instead of ML, and have been applied to synthetic chemistry problems.[35]

1.4 How Is It Done?

Execution of ML research for chemistry and materials commonly leverages the "scientific Python" stack, including numpy, pandas, scikit-learn, and additional neural network packages such as TensorFlow and Keras. We endorse use of that stack, although languages such as R, C++, and Julia are also used in this field. The following guides are concise instructions for self-education to begin performing work using those tools for ML for materials, in particular, molecular materials. It is assumed that the motivated individual has some (perhaps limited) experience in programming and needs to improve their Python skills. The guides heavily leverage disparate public resources such as tutorials on the Web and academic publications, but prescribe a course of study that will allow someone to begin research in this field within a minimal amount of time: 1–4 weeks depending on level of experience, not including complete reading of recommended books. Further, the final exercises expose the reader to important concepts typically taught in an undergraduate computer science course on algorithms. If you work with DOD and are unable to retrieve any of the resources mentioned in the guides, please contact the author. After completion of these guides, including reading recommended articles and relevant parts of recommended books, a researcher should generally be able to understand and conceive how to reproduce publications on ML in the chemistry and materials science literature.

2. Bootstrap Yourself into Machine Learning for Materials

1. Install Anaconda (Python 3 version) on your computer of choice. It is useful to install this on a laptop/desktop to use features that couple to your Web browser. Anaconda is free and available at https://www.anaconda.com/download/. Review its documentation and launch Spyder. Anaconda is available as a computational science environment module on DOD supercomputers.

2. (multiday task) Familiarize yourself with the Python language, its standard library, and third-party modules used for science. Completing the tutorial at https://docs.python.org/3/ is essential; browsing the library/language references is useful. When learning Python and studying example code, you should execute that code in Spyder and reproduce the results. Reproducing results ensures that the example is correct for your version of Python (not always the case) and helps you "learn by doing". Work the examples in Sections 1.1 through 2.1 at http://www.scipy-lectures.org/. Browse Python style at https://www.python.org/dev/peps/pep-0008 to help with writing easily understood code.

3. Reproduce and understand every example at http://scikit-learn.org/stable/tutorial/index.html in the first two sections ("intro to ML" and "tutorial on statistical-learning"). Try to grasp "how" the examples work (data structures and application programming interfaces (APIs)). Your future research efforts will probably heavily leverage scikit-learn tools, so feel free to explore.

4. Understand what pandas is and how to get started using it. See http://pandas.pydata.org/pandas-docs/stable/10min.html and work the examples. When would you use pandas instead of just numpy?

5. Using the conda command line tool, make a new Anaconda environment *that includes the anaconda metapackage* and install TensorFlow in it. See: https://conda.io/docs/user-guide/tasks/index.html for managing environments and packages, and https://anaconda.org/anaconda/tensorflow for tensorflow.

6. Install RDKit in the same environment as tensorflow, again using the conda command line tool. See https://anaconda.org/rdkit/rdkit and http://www.rdkit.org/docs/GettingStartedInPython.html for details on installation and rdkit capabilities. Study and test the examples in the RDKit overview.

7. Fix and reproduce this example (make an equivalent model) using its data, TensorFlow's Keras, and RDKit: https://www.wildcardconsulting.dk/useful-information/molecular-neural-network-models-with-rdkit-and-keras-in-python/. Tip: use "from tensorflow.python.keras" as the syntax to import functions from keras and reformat the input data file as needed.

8. (multiday task) Execute the crash course at https://developers.google.com/machine-learning/crash-course/prereqs-and-prework including all programming exercises. You do not need to understand all of the TensorFlow syntax—just the main points. Many model construction and training techniques covered in this course transfer to a wide variety of other problems. Additional datasets to explore are available at websites such as https://www.kaggle.com or http://archive.ics.uci.edu/ml/index.php.

9. We recommend you read some landmark articles, including "Statistical Modeling: The Two Cultures"[28] and "To Explain or to Predict?".[27] We recommend the following books: *The Elements of Statistical Learning*, 2nd edition,[36] *Computer Age Statistical Inference*,[37] and *Python Data Science Handbook*.[38] US Army Research Laboratory staff may contact the author for those publications, and more.

3. Python Exercises: Validate Your Bootstrapping

Preamble: For any of these exercises, do not just Google the solution. That defeats the point. Limit yourself to checking language or library documentation. Most documentation is available via executing "help()" or "?" or "%quickref" at the ipython console prompt in Spyder. Use Python for all exercises.

1. Solve at least the first six problems at https://projecteuler.net/archives. The first six answers are: 233168, 4613732, 6857, 906609, 232792560, and 25164150.
2. http://codingbat.com/python is a site that will allow you to enter code to solve test problems directly at its website, and then execute your code to see whether it provides the output as requested by the test. Without looking at the solutions, correctly solve at least half of the problems in each section (Warmup-1, List-1, etc.), with the exception of String-2. You can skip String-2. CodingBat was chosen because it does not require any sort of account creation or login to use its examples. If you enjoy this format of online exercise, visit: https://leetcode.com/, https://www.hackerrank.com/, https://www.codewars.com, or https://coderbyte.com/. Those sites are better than CodingBat, but require you to create an account.
3. We now move on to exercises using ipython notebooks, also called Jupyter notebooks. ipython notebook will be installed by default if you are using Anaconda, as recommended. To start the Jupyter Notebook environment, do one (and only one) of the following things: 1) launch Jupyter Notebook from the Anaconda Navigator, 2) launch Jupyter Notebook from its Windows start menu shortcut, or 3) execute "jupyter notebook" from an Anaconda command-line prompt. After any of those three items, your default browser should open up a new window/tab showing a list of directories and files. If you have an ipython notebook (.ipynb file extension) in one of your directories, you will be able to open it through this interface. More information is available at https://jupyter.org/documentation and https://docs.anaconda.com/anaconda/user-guide/tasks/use-jupyter-notebook-extensions.
4. Using the "no_solution" variant of available ipython notebooks, complete at least 50 of the 100 numpy exercises at https://github.com/rougier/numpy-100/. If you get stuck, try viewing the "with_hint" notebook. Afterwards, view the full notebook to check your work and demonstrate efficient code.
5. "Challenge" notebooks covering advanced concepts and containing unit tests are available at https://github.com/donnemartin/interactive-coding-challenges. As an example, complete the fizz buzz challenge and view its solution notebook. Complete at least 10 of the following "challenge" notebooks: find the digital root, generate a list of primes, merge tuple ranges, determine if a string is a permutation, determine if a number is a power of 2, find the highest product of three numbers, implement a linked list, search a sorted matrix for an item, determine an island's perimeter, create a class supporting insert […] in O(1), find a unique number of ways to represent *n* cents given coins, maximize stock prices given *k* transactions, maximize items placed in a knapsack, implement a binary search tree, implement a graph, implement breadth-first search on a graph, implement depth-first search on a graph, implement a stack, implement a queue, implement a priority queue, and/or find the shortest path in a weighted graph. If you do not understand the statement of a problem, do external reading, ask a programmer friend, or skip it. Some graph problems require challenges to be completed in a specific order and may require use of supporting .py files available in the repo. Afterward, view the solution notebooks.

4. References

1. Lookman T, Alexander FJ, Bishop AR. Perspective: codesign for materials science: an optimal learning approach. APL Materials. 2016;4(5):053501.
2. Ryan S, Thaler S, Kandanaarachchi S. Machine learning methods for predicting the outcome of hypervelocity impact events. Expert Systems with Applications. 2016;45:23–39.
3. Mueller T, Kusne AG, Ramprasad R. Machine learning in materials science: recent progress and emerging applications. In: Parrill AL, Lipkowitz KB, editors. Reviews in computational chemistry. New York (NY): John Wiley & Sons; 2016; Vol. 29; Chapter 4.
4. John S. McCain National Defense Authorization Act for Fiscal Year 2019. H.R. 5515, 115th Cong. 2018. https://www.congress.gov/bill/115th-congress/house-bill/5515/text.
5. Association of the United States Army. CMF#1: Army Futures Command Unifies Force Modernization. 2018 AUSA; 2018 Oct 8–10; Washington, DC. https://www.ausa.org/events/2018-annual-meeting/sessions/cmf-1-army-futures-command-unifies-force-modernization.
6. Chen H, Engkvist O, Wang Y, Olivecrona M, Blaschke T. The rise of deep learning in drug discovery. Drug Discovery Today. 2018;23(6):1241–1250.
7. Pyzer-Knapp EO, Li K, Aspuru-Guzik A. Learning from the Harvard clean energy project: the use of neural networks to accelerate materials discovery. Advanced Functional Materials. 2015;25(41):6495–6502.
8. Ward L, Agrawal A, Choudhary A, Wolverton C. A general-purpose machine learning framework for predicting properties of inorganic materials. Computational Materials. 2016;2:16028.
9. Bostanabad R, Zhang Y, Li X, Kearney T, Brinson LC, Apley DW, Liu WK, Chen W. Computational microstructure characterization and reconstruction: review of the state-of-the-art techniques. Progress in Materials Science. 2018;95:1–41.
10. Moore BA, Rougier E, O'Malley D, Srinivasan G, Hunter A, Viswanathan H. Predictive modeling of dynamic fracture growth in brittle materials with machine learning. Computational Material Science. 2018;148:46–53.

11. Tabor DP, Roch LM, Saikin SK, Kreisbeck C, Sheberla D, Montoya JH, Dwaraknath S, Aykol M, Ortiz C, Tribukait H, et al. Accelerating the discovery of materials for clean energy in the era of smart automation. Nature Reviews Materials. 2018;3(5):5–20.

12. Kim E, Huang K, Jegelka S, Olivetti E. Virtual screening of inorganic materials synthesis parameters with deep learning. Computational Materials. 2017;3(1).

13. Li Z, Kermode JR, De Vita A. Molecular dynamics with on-the-fly machine learning of quantum-mechanical forces. Physical Review Letters. 2015;114(9):096405.

14. Mitchell TM. The discipline of machine learning. Pittsburgh (PA): School of Computer Science, Carnegie Mellon University; 2006 July. Report No.: CMU-ML-06-0108.

15. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, et al. Mastering the game of Go without human knowledge. Nature. 2017;550(7676):354–359.

16. Mosquera MA, Fu B, Kohlstedt KL, Schatz GC, Ratner MA. Wave functions, density functionals, and artificial intelligence for materials and energy research: future prospects and challenges. ACS Energy Letters. 2017;3(1):155–162.

17. Zhou Q, Tang P, Liu S, Pan J, Yan Q, Zhang SC. Learning atoms for materials discovery. Proceedings of the National Academy of Sciences of the USA. 2018;115(28):E6411–E6417.

18. Gilmer J, Schoenholz SS, Riley P, Vinyals O, Dahl GE. In neural message passing for quantum chemistry. Proceedings of the 34th International Conference on Machine Learning; 2017 Aug 6–11; Sydney, Australia. https://arxiv.org/pdf/1704.01212.pdf.

19. Jin W, Coley CW, Barzilay R, Jaakkola TS. In predicting organic reaction outcomes with Weisfeiler-Lehman Network. Proceedings of the 31st Conference on Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA.

20. Ramprasad R, Batra R, Pilania G, Mannodi-Kanakkithodi A, Kim C. Machine learning in materials informatics: recent applications and prospects. Computational Materials. 2017;3(1).

21. Wu Z, Ramsundar B, Feinberg EN, Gomes J, Geniesse C, Pappu AS, Leswing K, Pande V. MoleculeNet: a benchmark for molecular machine learning. Chemical Science. 2018;9(2):513–530.

22. Zhang, Y, Ling C. A strategy to apply machine learning to small datasets in materials science. Computational Materials. 2018;4(1).

23. von Lilienfeld OA. Quantum machine learning in chemical compound space. Angewandte Chemie International Edition. 2018;57(16):4164–4169.

24. Granda JM, Donina L, Dragone V, Long DL, Cronin L. Controlling an organic synthesis robot with machine learning to search for new reactivity. Nature. 2018;559(7714):377–381.

25. Rupp M. Machine learning for quantum mechanics in a nutshell. International Journal Quantum Chemistry. 2015;115:1058–1073.

26. Sanchez-Lengeling B, Aspuru-Guzik A. Inverse molecular design using machine learning: Generative models for matter engineering. Science. 2018;361(6400):360–365.

27. Shmueli G. To explain or to predict? Statistical Science. 2010;25(3):289–310.

28. Breiman L. Statistical modeling: the two cultures. Statistical Science 2001;16(3):199–231.

29. Breiman L. Random forests. Machine Learning. 2001;45:5–32.

30. Rasmussen CE, Williams CKI. Gaussian processes for machine learning. Cambridge (MA): MIT Press; 2006.

31. Smola AJ, Scholköpf B. A tutorial on support vector regression. Statistics and Computing. 2004;14:199–222.

32. Segler MHS, Preuss M, Waller MP. Planning chemical syntheses with deep neural networks and symbolic AI. Nature. 2018;555(7698):604–610.

33. Hofmann T, Schölkopf B, Smola AJ. Kernel methods in machine learning. The Annals of Statistics. 2008;36(3):1171–1220.

34. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436–444.

35. Zhou Z, Li X, Zare RN. Optimizing chemical reactions with deep reinforcement learning. ACS Central Science. 2017;3(12):1337–1344.

36. Hastie T, Tibshirani R, Friedman J. The elements of statistical learning. 2nd ed. New York (NY): Springer; 2016.

37. Efron B, Hastie T. Computer age statistical inference: algorithms, evidence, and data science (Institute of Mathematical Statistics Monographs). New York (NY): Cambridge University Press; 2016.

38. VanderPlas J. Python data science handbook. Boston (MA): O'Reilly Media; 2016 Nov.

List of Symbols, Abbreviations, and Acronyms

2-D	2-dimensional
3-D	3-dimensional
AI	artificial intelligence
CL-20	hexanitrohexaazaisowurtzitane
DOD	US Department of Defense
ML	machine learning
R&D	research and development

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIR ARL IMAL HRA RECORDS MGMT RDRL DCL TECH LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	DTRA R AND D D DALTON
1 (PDF)	ONR CODE 35 C STOLTZ
1 (PDF)	US ARMY RDECOM M HOHIL
1 (PDF)	US ARMY RDECOM ARDEC A DI STASIO
1 (PDF)	USAF AFMC AFOSR/RT M SCHMIDT
1 (PDF)	USAF AFMC AFRL/RW S PEIRIS
1 (PDF)	USAF AFMC AFRL/RWME C LINDSAY
19 (PDF)	ARL RDRL WM A RAWLETT J ZABINSKI RDRL WML T SHEPPARD W OBERLE RDRL WML B B BARNES B RICE E BYRD N TRIVEDI RDRL WML C J SADLER RDRL WML D R BEYER RDRL WML E F FRESCONI RDRL WML F M ILG RDRL WML G M MINNICINO RDRL WMM B B LOVE RDRL WMM C J SNYDER RDRL WMM G B RINDERSPACHER RDRL WMP D LYON RDRL WMP D R DONEY RDRL WMP G B ROOS